How Good are Association-rule Mining Algorithms ?

Vikram Pudi [*]   Jayant R. Haritsa [*]

1. Introduction

We address the question of how much space remains for performance improvement over current association rule mining algorithms. Our approach is to compare their performance against an **"Oracle algorithm"** that knows *in advance* the identities of all frequent itemsets in the database and only needs to gather the actual supports of these itemsets, in one scan over the database, to complete the mining process. Clearly, *any* practical algorithm will have to do at least this much work in order to generate mining rules.

While the notion of the Oracle is conceptually simple, its *construction* is not equally straightforward. In particular, it is critically dependent on the choice of data structures and database organizations used during the counting process. We present a carefully engineered implementation of Oracle that makes the best choices for these design parameters at each stage of the counting process.

We also present a new mining algorithm, called **ARMOR** (Association Rule Mining based on ORacle), whose structure is derived by making minimal changes to Oracle, and is guaranteed to complete in two passes over the database. This is in marked contrast to the earlier approaches which designed new algorithms by trying to address the limitations of *previous* online algorithms. Although ARMOR is derived from Oracle, it shares the positive features of a variety of previous algorithms such as PARTITION, CARMA, AS-CPA, VIPER and DELTA. Our empirical study shows that ARMOR consistently performs within a *factor of two* of Oracle, over both real and synthetic databases.

2. The Oracle Algorithm

The Oracle algorithm takes as input the database $\mathcal{D}$ in item-list format, the set of frequent itemsets F and its negative border N, and produces as output the supports of itemsets in $F \cup N$. A partitioning strategy is used to determine these supports. Counters of singletons and pairs are stored in direct lookup arrays, whereas a DAG data-structure is used for counting longer itemsets. A judicious combination of tid-lists and bit-vectors are used to ensure efficiency in the counting process.

We prove that Oracle is optimal in that only required itemsets are enumerated at any stage and the cost of enumerating each itemset is $\Theta(1)$. Our experimental results show that there is a substantial gap between the performance of current mining algorithms such as VIPER and FP-growth and that of Oracle.

3. The ARMOR Algorithm

Our new online algorithm, ARMOR, differs from Oracle only in that it also includes candidate generation and removal processes during its first pass. Techniques from *incremental* mining algorithms are used in order to generate and remove candidates efficiently. The partitions scanned so far are treated as the "original database" and the current partition is viewed as the "increment", thereby not requiring mining each partition from scratch.

In the second pass, which involves only candidate removal, but no generation, the counts of candidates remaining at the end of the first pass are determined. Since our partitioning strategy ensures that the counts of most candidates are available at the end of the first pass itself, the second pass is typically "light and short".

An extensive empirical performance evaluation shows that ARMOR performs within a factor of two of the Oracle, with regard to response time, on both real and synthetic databases, with acceptable main memory utilization. These performance characteristics arise because ARMOR utilizes the same counting techniques as that of Oracle and the number of itemsets it processes in each of its two passes are comparable to what Oracle processes in its single pass.

The complete details of the issues involved in the design, implementation and evaluation of the Oracle and ARMOR algorithms are available in the full version of this paper [1].

References

[1] V. Pudi and J. Haritsa. On the optimality of association-rule mining algorithms. Technical Report TR-2001-01, DSL, Indian Institute of Science, 2001.

[*]Database Systems Lab, SERC, Indian Institute of Science, Bangalore 560012, INDIA. Email: {vikram, haritsa}@dsl.serc.iisc.ernet.in